Determining the timing of project control points using a facility location model and simulation

Narjes Sabeghi[1,2], Hamed. R. Tareghian[1], Erik Demeulemeester[2], Hasan Taheri[3]

[1] Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, Mashhad, Iran
[2] Research Center for Operations Management, Faculty of Business and Economics, Katholieke Universiteit Leuven, Leuven, Belgium
[3] Khayyam University, Mashhad, Iran

Abstract

Projects are usually performed in relatively unstable environments. As such, changes to the baseline schedules of projects are inevitable. Therefore, project progress needs to be monitored and controlled. The control process can be assumed as a continuum in which one side is continuous control and the other side is no-control. Continuous control and no-control strategies are cost-wise prohibited. Hence, project progress should be controlled at some discrete points in time during the project's duration. The optimal number and timing of control points are the main issues that should be addressed. In this paper, taking a *dynamic view* to the project control, for the first time we use an adapted version of the facility location model (FLM) to find the optimal timing of project control points. Initially, the adapted FLM determines the optimum timing of the control points in the project's duration. A simulation model is then used to predict the possible disruptions in the time period between the beginning of the project and the first control point (*monitoring phase*). If no disruptions are observed, the project's progress is monitored in the second control point, otherwise possible corrective actions are taken using an activity compression model. Whenever due to disruptions, the baseline schedule is to be updated, the FLM is run again to determine the new timing of the control points for the rest of the project's duration. In an iterative manner, this process continues until the timing of the last control point is determined.

Keywords: Project control, Facility location model, Simulation.

1. Introduction

Project success is measured as the ability to complete the project according to the desired specifications, within the specified budget and according to the specified time schedule. However, rarely does a project finish with the same project plan as established in the final stage of the planning phase. Changes to the baseline schedule of projects seem to be inevitable. To complete projects successfully both planning and execution need to be properly implemented. In the absence of a formal process for reviewing and evaluating baseline schedule diversions, the resulting impact will be uncontrolled scope variance. The dynamic environment in which the majority of the projects are performed calls for dynamic control processes. In dynamic approaches, adjustments to the baseline schedule are taken as and when required. As a result, the baseline schedule may change and may require some rescheduling. One objective of the control process may be to minimize the deviations from the baseline schedule. The control process can be assumed as a continuum in which one side is continuous control and the other side is no-control. Continuous control may be the most effective type of project

control. However, it is cost-wise prohibitive. Implementing a no-control strategy may also be costly due to the possible penalties imposed on late delivery of the project and other losses due to not being able to deliver the project within the specified criterion. Therefore, a project's progress needs to be controlled at some discrete points in time during the project's duration. The timing of these discrete points (control points) can be specified and fixed prior to the start of the project (*static view*). However, in a dynamic view to control, the timing of control points can be changed during the execution of the project according to the state of the schedule.

In general, there are two approaches to deal with the uncertainty that stems from the dynamism inherent in the scheduling environment, namely proactive and reactive scheduling [6]. Proactive scheduling relies on the statistical knowledge of uncertainty and builds schedules that are less sensitive to project disruptions. Reactive scheduling involves revising a baseline schedule when an unexpected event occurs. In reactive scheduling one may reschedule when schedule diversions occur, either by completely regenerating a new schedule or by repairing an existing baseline schedule. In the current study, the latter view is taken.

In this paper and in the context of reactive scheduling with a repair strategy, for the first time we adapt a facility location model (FLM) to our purpose of finding the timing of control points. Our solution procedure consists of a computer simulation model combined with an adapted FLM as well as a project crashing model. After determining the first control point using the adapted FLM, the advancement of the project is simulated to predict the types and the magnitudes of deviations from the baseline schedule. In the next phase, using the project crashing model, the necessary adjustment steps are taken (repairing the schedule) to bring the project in line with the baseline schedule as much as possible. In so doing, our objective is to adjust the deviated schedule as soon as possible and also to increase the possibility of meeting the project's due date. The next control points are determined in an iterative manner.

The outline of the paper is as follows. In Section 2, some of the research articles that have dealt with the subject of project control are discussed and where appropriate the commonalities and differences to the current article are mentioned. The proposed method for project control is detailed in Section 3. The results regarding the validity of the method and its performance are given in Section 4. Section 5 concludes by presenting a summary of our study and also provides directions for future research.

2. Literature review

The development of a suitable control system is an important part of the project management effort. Furthermore, it is widely recognized that planning and monitoring play a major role as the cause of project failures. There have been a number of articles, e.g. [1, 3, 5, 7, 13, 14], published to support the importance of control in the achievement of project objectives. It has been shown that project performance can be improved if appropriate project control systems are in place. Some study by independent project analysis (IPA) identified that an optimal project control approach can reduce the execution schedule slip by as much as 15% [20].

In the following, we briefly discuss some of the research articles that deal with the subject of project control. We categorize our reviews as those articles that deal somehow with the determination of the optimal timing of control points and those articles that deal with more general aspects of project control.

Note that, because the novel part of the current study (determining the timing of control points) falls in the first category, where appropriate, we elaborate on the commonalities and differences to our present research.

The first category includes the following articles.

Partovi and Burton [13] carry out a computer simulation to compare the effectiveness of five control timing policies. The policies considered are no monitoring and control, monitoring and control at equal intervals, end-loaded (which advocates less intensive reviews in the early stages and more frequent reviews towards the completion of the project), front-loaded (which assumes more frequent reviews in the early stages and less reviews towards the completion of the project) and completely random monitoring. The comparison is made with respect to the amount of overrun time and also the amount of crashing effort they require in

controlling the project. The results indicate that although there are no significant differences among the policies in the amount of crashing effort spent, the end-loaded policy performs best in preventing time overruns. Note that, in contrast to Partovi et al. who are more concerned with the timing of control points under their five pre-defined control policies, we determine the timing of control points dynamically using a weight function that can be easily utilized to define any types of control policies including those studied by Partovi et al.

Tareghian et al. [16] use *simulation-optimization* to find the optimal number of control points as well as their timing. They use an evolutionary approach to determine the optimal number of control points and implement the so-called *electro-magnetism* in order to expedite the simulation process and to minimize the running cost. Based on a small sample of only 5 randomly generated projects with complexity indices ranging from 5 to 9, they conclude that the number of control points has an upper bound. In addition, in contrast to the results of Partovi et al., Tareghian et al. show that in the context of their studies, it is more beneficial to place the control points in the early stages of the project's duration. This may be due to the differences in the topology of the networks used in their study.

De Falco and Macchiaroli [3] propose a model for the quantitative determination of the timing of control points. Their approach is based on an effort function which is defined as a non-linear function of the total number of activities that are active at each time interval as well as the total slack time. By quantitative analysis of the effort function, they allocate appropriate control activities throughout the project's duration.

Raz and Erel [14] determine the optimal timing of project control points based on maximizing the amount of information generated by the control points. They describe the amount of information as a function of the intensity of the activities carried out since the last control point. The intensity of the activities being executed at any instant of time during the project's life cycle is determined using typical progress *s-curves*. They develop an optimal solution procedure based on dynamic programming and for a given number of control points, determine the timing of each control point. In contrast to Raz and Erel, in the current research a dynamic view to the project control is employed. However, similar to the *reporting delay* used by Raz and Erel to refer to the amount of time elapsed since the moment the activity took place, we utilize weighted distances in our method to force the timing of control points nearer to the heavily weighted potential control points (see Section 3).

Golenko-Ginzburg and Laslo [5] deal with the problem of production control in a semi-automated production system. They determine the next control point via simulation utilizing a constant time step. Referring to [5], a somewhat dynamic view to the determination of control points is taken. That is, with the objective of minimizing the number of control points (maximizing the time span between two adjacent control points), at any routine control point, given planned amount of production, planning horizon, actual accumulated amount of production observed at that control point and a chance constraint, the timing of the next control point is determined. Our approach differs in at least two fundamental aspects with the study of Golenko-Ginzburg and Laslo. Firstly, for the sake of convergence, they consider a minimal pre-given time span between two consecutive routine control points. We determine the control points (in our study, we call them potential control points) according to the structure of the project network which better reflects the high risk sections of the project that need more attention. Secondly, when some disruptions occur at a control point and the volume of production observed at that point is below the planned trajectory, they simply adjust the plan by connecting a straight line between the current position and the target position. In our approach, we utilize a crashing model to select the most appropriate combination of activities to be compressed so that the observed delays are possibly adjusted. In addition, every time the baseline schedule is modified to adjust the disruptions, the FLM determines the new timing of control points for the rest of the project, in the light of the current modifications.

The following articles may be classified in the second category. Kogan et al. [7] develop and solve a basic model for determining the optimal amount of control

effort that should be invested throughout the life cycle of homogenous projects in a deterministic environment. Homogenous projects are defined as projects having a large number of relatively similar activities. The model accounts for changing levels of project activity and includes two parameters to represent the effectiveness of the work management and control efforts. The objective is to trace project execution as closely to the planned intensity as possible while minimizing the cost of the control effort and also the cost of losses due to deviations from the plan.

Zhu et al. [19] study the problem of how to react when an ongoing project is disrupted. They propose a classification scheme for various types of disruptions and then define a recovery plan. The objective of their recovery plan is to bring the project on track with minimum cost. Their cost function comprises various costs related to the amount of deviation from the original plan. The problem is then formulated as an integer linear programming and is solved with a hybrid mixed-integer programming/constraint programming procedure which exploits a number of special features in the constraints.

Borrowing ideas from the statistical control process, Bowman [1] determines an upper control limit for each activity's duration. If during the execution of the project, the duration of an activity is considered to fall above the specified limit, a penalty is imposed and the point is considered within the limit. The objective of Bowman's approach is to determine the optimum upper limit for each activity's duration such that the control cost and penalties imposed are minimized while the probability of delivering the project on time is maximized. Using simulation, in an iterative method the optimum specification limits of the activity durations are obtained.

Van de Vonder et al. [17] evaluate the performance of various predictive-reactive project scheduling procedures with the objective of maximizing the schedule stability as well as the timely project completion probability. By predictive scheduling they mean a workable baseline schedule generated under the assumption that the environment in which the project is going to be executed is both deterministic and static. By reactive scheduling they mean the use of a set of procedures that are implemented when a schedule breakage occurs during project execution. They propose a proactive heuristic that aims at generating stable baseline schedules.

The well-known resource-constrained project scheduling problem (RCPSP) is used by Kuster et al. [9] as a conceptual framework to deal with the schedule disruptions in the context of an airport ground processes. Based on real examples taken from the domain of aircraft turnaround management, it is demonstrated that in order to optimally recover from a disrupted schedule, it is not enough to resort solely to the standard rescheduling techniques, but it is also necessary to consider alternative process execution paths. Considering that in many situations disruptions need to be dealt with in near-real-time, a genetic algorithm (GA) is developed and proposed for incremental schedule optimization. Appropriate initialization, crossover and mutation operators are also devised for the proposed GA.

Vanhoucke [18] proposes two different project tracking methods, namely top-down tracking and bottom-up tracking and compares their efficiencies. The top-down project tracking method is based on the parameters of the earned value management technique. The project performance data such as SV(t) and/or SPI(t) are used as early warning signals and trigger the need for corrective actions. The bottom-up tracking method relies mainly on schedule risk analysis. Schedule risk analysis yields sensitivity information with regards to each and every activity of the project and assumes that the focus should lie on only the highly sensitive parts of the project. The integration of his results in the new software tool (ProTrack) has led to the creation of a so-called "ProTrack's Assistant" which guides the user towards the best project control approach.

Deblaere et al. [2] formulates a reactive scheduling problem for the multi-mode RCPSP. Given a baseline schedule and a disruption with regards to resources or activities that occur during the execution of the baseline schedule, and with the objective of minimizing the rescheduling costs comprising of mode switching costs and costs incurred due to activity start time deviations, a reactive schedule is generated.

3. Methodology

Progress monitoring and control is vital for the success of projects. However, the controlling process itself is costly. Therefore, to optimize project control costs and at the same time ensure the project's success, one fundamental question needs to be considered, i.e. "how to determine the optimal timing of control points for a project in dynamic environments?" The purpose of this paper is to address this question. It seems logical to position control points as near as possible to the highly sensitive parts of the project and also at critical times throughout the project's duration. In some of the studies that were reviewed in Section 2, some criteria were proposed to distinguish these critical time points; for instance, De Falco and Macchiaroli [3] proposed a methodology to determine the timing of control points for projects based on an effort function, defined as a non-linear function of the total number of active operations and the total slack time. In the current study, we associate a weight with each project activity based on the degree of its importance and criticality and then we apply the adapted FLM to determine the timing of control points dynamically. A simulation model carries the advancement of the project and also predicts possible disruptions. A crashing model is used to possibly adjust the schedule whenever some diversions from the baseline schedule are observed. The adaptation process of the FLM to our purpose together with the structure of the adapted model is presented in the next subsection. In the second subsection, two methods (all point control and simulated annealing) are introduced which are primarily used as a means to demonstrate the validity of the proposed method. Moreover, we use 48 instances from each of the well-known J30, J60, J90 and J120 sets of PSPLIB [8] to show the effectiveness of our method in dynamically controlling the progress of these standard project instances. These projects are subjected to some disruptions and the objective of the control is to complete them as close as possible to their original due dates (see Section 4).

3.1. Proposed approach

3.1.1. Facility location model (FLM)

Facility location is related to locating or positioning at least one new facility among several existing facilities in order to optimize (minimize or maximize) at least one objective function (i.e. cost, profit, revenue, travel distance, service waiting time, coverage or market shares).

In an ideal project control process, a project's progress is monitored continuously and reactive actions are taken immediately upon the occurrences of disruptions. However, in real-world projects this might not be practical and may also be cost-wise prohibitive. Instead, an effective project control process focuses on critical aspects of the project along its life cycle and if required, takes appropriate adjustments as soon as possible. Practically, this may lead to considering some discrete points (*potential control points*). These points are being chosen along the life cycle of a project based on their anticipated importance and criticality. Then the aim is to select a subset of potential control points from them such that the weighted distance between the selected points (*actual control points*) and the potential control points is minimized. In other words, the time locations of the actual control points are as close as possible to the potential control points. Paying closer attention, it becomes clear that this situation can be adapted to the min-sum facility location model. Potential control points of the adapted FLM play the role of the existing facilities of the original FLM. It is possible to consider different options for the definition of potential control points. For instance, one can set the potential control points at the end of every week or at the end of every month. However, when potential control points are set at prefixed intervals (weekly, etc.), at those time instances some project activities may have completed only a small fraction of their durations. Therefore, a meaningful control may not be achieved. In other words, the information gathered at the completion of the activities (scheduled completion times) that are being monitored, provide more accurate information concerning the deviation in the durations and puts the project manager in a better position to evaluate the progress of the project and make better decisions with respect to the possible adjustment scenarios, etc.

Hence, we have considered the time instances at which one or more activities of the project end their executions according to the baseline schedule, as the potential control points (existing facilities).
In the original FLM, a weight is associated with each of the facilities and in min-sum location models the goal is to minimize the sum of the weighted distances from the new facilities. One can think of many different ways for associating a weight to each potential control point that indicates the degree of its suitability or importance to be selected as a control point. For example, one can consider the volume of the work being finished at each control point or the total number of successors of the finished activities or the activities' intensity distribution [14] or even the criticality index [4] as a measure of weight associated with that point. In this paper, we initially assign a weight ω_i to each activity $i \in A$ of the project, where A is the set of project activities. As it can be seen from Equation (3.1), the activity weights are considered to be a function of c_i and f_i, $i \in A$, where c_i is the number of the critical paths that contain activity i and f_i is the total float of non-critical activity i. The weight of critical activity i has a direct relation to the number of critical paths of the project network; in other words, as Elmaghraby [4] states, if there are 10 critical paths in the network and activity a lies on all of them while activity b lies on only 4, then activity a is pronounced of 'higher criticality' than activity b. For non-critical activities the amount of their total float is used to indicate the degree of their importance (control-wise); i.e., if the total float of activity a is more than the total float of activity b, then the weight of b should be more than the weight of a, because delays in activity b have probably more effect on the project's due date. So the relationship between the total float and the weight is an inverse relationship. Therefore, the weight ω_i is defined as follows:

$$\omega_i = \begin{cases} 2^{c_i} & \text{if } i \text{ is critical} \\ 2^{-f_i} & \text{if } i \text{ is non-critical} \end{cases} \tag{3.1}$$

The relation implies that the weight ω_i is directly dependent upon the number of critical paths that contain activity i and inversely related to the total float of non-critical activity i (note that an activity $i \in A$ is either critical or non-critical). We use an exponent of 2 to strengthen the role of the critical activities on the critical paths and also to weaken the role of the activities that have larger values of total float. To illustrate, consider an activity that lies on only 1 critical path and compare it with another activity that lies on 2 critical paths. Using (3.1), the associated weights would be 2 and 4, highlighting the role of the critical activities that lie on more critical paths. In contrast, as expected the weights assigned to the non-critical activities that have larger values of total floats by (3.1), are closer to zero. Therefore, a non-critical activity with a total float of 10 would have a much smaller weight than an activity with a total float of 5. Note that Equation (3.1) gives one possible way to compute the weight ω_i. Obviously, one could also have used a different relationship, provided that ω_i is directly related to c_i, and inversely related to f_i.
Next, we determine W_j, the weight associated with potential control point j, as follows:

$$W_j = a_j \sum_{i \in EA_j} \omega_i \tag{3.2}$$

in which EA_j (*Effective Activities*) is the set of all activities finishing at control point j as well as their immediate successors. Indeed, the associated weight for existing points (potential control points) is determined according to the effect of the deviations of these points on the project duration. These possible deviations are the result of delays that have occurred either during the performance of the activities that end at that point or prior to it. We do not have complete information about the deviations of the activities that are still in progress at these points and if there are some deviations from their scheduled finish times, it will be considered at the existing point corresponding to their scheduled finish time. Parameter a_j is the possible amount that the remaining part of the project (from time instant j to the end of the project) can be compressed. Noting that the network paths that are currently non-critical may become critical after critical paths are compressed (crashed), a_j is determined by summing up the amount of durations that critical paths can be possibly crashed. Let $\Pi = \{\pi_p, p = 1, 2, ..., P\}$ be the set of all network paths. Suppose $\pi_1, \pi_2, ..., \pi_l$ are the critical paths with lengths lcp and $\pi_{l+1}, \pi_{l+2}, ..., \pi_P$ are the non-

critical paths with lengths $l(\pi_p), (p = l{+}1, l{+}2, ..., P)$. Then the maximum amount of project compression that is possible from point j on is calculated as follows:

$$\begin{aligned} pct_j = \min_{\Pi}\{\{ & \sum_{k\in\pi_p \ \&\ st(k)\geq j} (d_k - cd_k) \\ & , p = 1, 2, ..., l\}, \\ \{ & \sum_{k\in\pi_p \ \&\ st(k)\geq j} (d_k - cd_k) + \\ & (lcp - l(\pi_p)), p = l+1, ..., P\}\}. \end{aligned} \tag{3.3}$$

In (3.3), d_k and cd_k are the expected and crashed duration of activity k respectively and $st(k)$ is the start time of activity k. To balance the scale of the coefficients, we normalize pct_j by dividing it by pct_{max} ($pct_{max} = \max_{j=1,...,m}(pct_j)$, where m is the number of potential control points). Now, we calculate a_j as follows:

$$a_j = pct_j / pct_{max} \tag{3.4}$$

To clarify how pct_j is determined, we present the following small example.

Example 3.1. *Suppose that 10 units of time have elapsed since the start of a project. This project's network contains three paths, one of which is currently critical. The lengths of the network's paths and also the amount by which we can reduce them are given in Table 1. Now* pct_{10} *is calculated as:*

$$pct_{10} = \min(4, 1 + (7-5), 2 + (7-3)) = 3$$

It is clear that it is no use crashing the critical path with 4 units of time as path 2 can only be crashed to a length of $5 - 1 = 4$ *time units.*

Table 1: Example for computing pct_{10}

Path	Current Length	Possible Compression
1	7	4
2	5	1
3	3	2

The weight function as defined in (3.2) assigns higher weights to potential control points where the degree of dependency of the rest of the project on activities that finish at that point is high, and at the same time the opportunity to adjust for any possible deviations that may occur in the baseline schedule is high as well. Unlike other studies, e.g. Partovi and Burton's work [13], in our proposed approach, the weight function plays a crucial role in selecting the control points based on the project's parameters such as the precedence relations between activities, the non-critical activities' total floats, etc. In Appendix A, a small example is given to illustrate how the locations of control points are changed as the weight function changes.

To formulate the FLM, we consider y_j and x_{ij} as decision variables. These are defined as follows:

$$y_j = \begin{cases} 1 & \text{if the } j^{th} \text{ potential control point is} \\ & \text{actually selected to be a control point} \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ potential control point} \\ & \text{is to be controlled at the } j^{\text{th}} \\ & \text{control point} \\ 0 & \text{otherwise} \end{cases}$$

By controlling the i^{th} potential control point at the j^{th} control point, we mean monitoring the progress of the project and adjusting for any possible deviations that have occurred up to the i^{th} potential control point at the j^{th} control point. These possible deviations are the result of delays that have occurred either during the performance of the activities that end at that point or prior to it. We control project's progress from the last control point up to point i at potential control point j, only if j is selected to be a control point, that is,

$$x_{ij} \leq y_j \qquad \text{for all } i \text{ and } j \tag{3.5}$$

Also, we add another constraint to ensure that every potential point is controlled only by one of the control points. It is obvious that we cannot adjust the schedule for the deviations of the activities that finish at the i^{th} potential control point before this point, therefore we have,

$$\sum_{j\geq i} x_{ij} = 1 \qquad \text{for all } i \tag{3.6}$$

As mentioned previously, it is advisable to monitor the project progress at a limited number of points during its duration. The number of actual control points in a project, n, can be specified according to e.g. the number of milestones, the amount of control budget or as a management decision. Assuming that there are m potential control points, then we have

$$\sum_{j=1}^{m} y_j \leq n \tag{3.7}$$

In the adapted FLM, the objective function is $\sum_{i=1}^{m}\sum_{j=i}^{m} W_i d_{ij} x_{ij}$ which minimizes the sum of the weighted distances. Note that, d_{ij} is the time distance between the i^{th} and the j^{th} potential control points. Therefore, the proposed adapted FLM for determining the timing of control points is as follows:

$$\begin{aligned}
(FLM) \quad & \min \sum_{i=1}^{m}\sum_{j=i}^{m} W_i d_{ij} x_{ij} \\
& s.t. \\
& x_{ij} \leq y_j && i = 1,...,m, \\
& && j = 1,...,m, \\
& \sum_{j=i}^{m} x_{ij} = 1 && i = 1,...,m, \\
& \sum_{j=1}^{m} y_j \leq n, \\
& x_{ij} \in \{0,1\} && i = 1,...,m, \\
& && j = 1,...,m, \\
& y_j \in \{0,1\} && j = 1,...,m.
\end{aligned}$$

This model is a special case of min-sum models, namely the *n-median* problem [11]. There are some approaches for solving this model. In this paper, we use a linear programming relaxation by relaxing the constraint $y_j \in \{0,1\}$ to

$$y_j \geq 0. \tag{3.8}$$

It can be shown that the LP relaxation always gives an optimal solution for the class of simple networks like *line networks* (a line network consists of a single simple path) [11, 12]. Because the network that we will have in project control problems is a line network (a single path consisting of the potential control points), we can use the LP relaxation to obtain the optimal solution of the adapted model.

3.1.2. Simulation model (SM)

After solving the FLM, we will have n control points $cp_1, cp_2, ..., cp_n$. In the next stage, a simulation model is used to predict possible deviations from the baseline schedule. The simulation is carried out between the corresponding control points, namely $[0, cp_1], [cp_1, cp_2], ..., [cp_{n-1}, cp_n]$. The possible deviations are dealt with at each control point using activity compression. If no deviation is observed in an interval, the next control point is considered and the time interval is simulated. In this paper, we implement a system world-view, called the *discrete-event simulation* paradigm [10]. After adjusting the deviations and updating the project network, we will continue with new points and new weights. Algorithm 1 explains the steps taken by the simulation model (we refer the reader to [10] regarding some basic subjects such as random sampling from distributions, etc. that we have referred to in Algorithm 1). This algorithm presents steps necessary to simulate the advancement of the project at $[cp_{i-1}, cp_i]$, $i = 1, 2, ..., n$, where $cp_0 = 0$ (the start point of the project).

3.1.3. Crashing model (CM)

Deviations from the baseline schedule can be dealt with in various ways using *disruption management* methods. In this paper, at each control point when some deviations from the baseline schedule are observed, an activity compression model is used to possibly expedite the remainder of the project (from the current control point to the end of the project) and to bring back the deviated schedule as close as possible to the baseline schedule. Of course, the amount of adjustments (repairing the diverted schedule such that it becomes as close as possible to the baseline schedule) depends on the possibility of expediting the set of activities that have not started their execution yet. The objective of the crashing model used in this paper is different from the classical time/cost trade-off

Algorithm 1 Simulation Model

Input:
Simulation interval $[cp_{i-1}, cp_i]$;
the activities' duration distribution;
the activities' scheduled start and finish time sst and sft and the number of activities q;
Output:
new planned start and finish times after updating the project network;
begin

$T \leftarrow cp_{i-1}$ (T: simulation time).
for $j = 1$ **to** q **do**
  $status_j \leftarrow 0$ (status is a variable that is 1 when the activity has started).
end for
determine the actual finish time (aft) of each activity after randomly sampling from its duration distribution [10].
while there are some activities whose sft is between cp_{i-1} and cp_i and $T \leq cp_i$ **do**
  for all activities whose sst is less than or equal to T and whose $status = 0$ **do**
    $status_j = 1$.
    if $aft_j \leq cp_i$ **then**
      $sft_j \leftarrow aft_j$;
    else
      $sft_j \leftarrow cp_i + 1$.
    end if
    update the network.
  end for
  $T \leftarrow T + 1$.
end while

end

problem. Every time that crashing model is employed to compensate for the project's delays, initially a set of critical activities that can be compressed is established. Then the model selects an activity from this set that has a start time which is as near as possible to the time of the current control point. In this way, in addition to trying to repair a disrupted schedule, the opportunity for crashing activities (should the need arise) later on in the duration of the project is not taken away. The activity compression model is presented in Algorithm 2. First, let us introduce some notations that we use in the description of Algorithm 2:
cpath(i): i^{th} critical path,
ncpath(k): k^{th} non-critical path,
lcp: the length of the critical path,
l(p): the length of path p,
ca(i,j): j^{th} activity on cpath(i),
pa(i): the activity that is chosen on cpath(i) for crashing,
d(j): expected duration for activity j,
cd(j): crashed duration for activity j,
sft(j): scheduled finish time for activity j (dummy end activity is denoted by q),
u(i): the amount of crashing that is possible for cpath(i):

$$u(i) = \min\{\min_{\{k:pa(i)\notin ncpath(k)\}}\{lcp - l(ncpath(k))\}, d(pa(i)) - cd(pa(i)), sft(q) - duedate\}. \quad (3.9)$$

maxc: current maximum possible crashing ($maxc = \min(u(i) : \forall i)$
The compression of the critical path(s) length should

Algorithm 2 Crashing Algorithm

Input:
the activities' scheduled finish time sft;
the critical and non-critical paths ($cpath$ and $ncpath$) and their corresponding lengths (lcp and $l(ncpath)$);
the project due date ($duedate$);
Output:
new planned start and finish times after updating the project network and the project finish time (C_{max}) that is as close as possible to $duedate$;
begin

while $sft(q) > duedate$ **do**
  for all critical paths cpath(i) **do**
    find pa(i) and compute $u(i)$.
  end for
  $maxc = \min(u)$.
  Reduce the duration of all selected $pa(i)$ (do crashing) ($d(pa(i)) = d(pa(i)) - maxc$).
  Update the network.
  Recompute the critical paths.
end while

end

always be done in the light of the length of the non-critical path(s) (see the first part of Equation (3.9)). In each step, after updating the paths' length, we recompute the critical paths and continue to reach the due date.

3.1.4. Integrated algorithm

Now, integrating the FLM, the SM and the CM, we present a methodology for dynamically controlling a project. To do this, initially we employ the FLM to determine the current control points. Note that, before applying the FLM, we should determine the potential control points $x_1, x_2, ..., x_m$ and compute their associated weights $(W_1, ..., W_m)$. We then simulate the advancement of the project for the interval $[0, cp_1]$ in order to predict the deviations. If no deviations are observed, we go to the next interval and simulate the advancement of the project for $[cp_1, cp_2]$, otherwise we apply the CM to adjust the deviations. Hence the project network is updated. Now, the FLM is applied to the updated network and the whole process is repeated. An overview of this methodology is illustrated in Algorithm 3.
To demonstrate the validity of the proposed integrated algorithm, we compare its performance to the performances of two alternative methods that we have developed specifically for this purpose. These two methods that are called all point control (APC) and simulated annealing (SA) [15] are described in the next section. The only difference in these three methods is the way of determining the control points. In the rest of the paper we refer to them briefly as FLM, APC and SA. By the FLM, we mean the whole process containing: determining the control points, simulation (SM) and compression (CM).

3.2. Alternative approaches

3.2.1. All point control (APC)

By all point control we mean controlling the project progress at each and every potential control point. Therefore, we initially determine the potential control points $x_1, x_2, ..., x_m$. Remember that these are time points during the project's duration at which at least one activity ends its execution according to the baseline schedule. Now there are m control points $cp_1 = x_1, cp_2 = x_2, ..., cp_m = x_m$. Next we use the SM and the CM to predict the possible deviations and to bring the project back on track, respectively.

Algorithm 3 Integrated Algorithm

Input:
the existing points $x_1, x_2, ..., x_m$;
the weights W_i for $i = 1, 2, ..., m$;
the number of control points (n);
the baseline schedule.
Output:
the project finish time that is as close as possible to *duedate*;
begin

1: $cp_0 \leftarrow 0, startpoint \leftarrow 1$.
2: $changelabel = 0$ (this is a binary variable which is used in the algorithm as follows:
if $changelabel = 0$ no need to recompute the timing of control points,
if $changelabel = 1$ the timing of control points need updating).
3: **while** $startpoint < m$ **do**
4: $p \leftarrow 1$.
5: construct the FLM and solve it to obtain control points $cp_1, ..., cp_n$.
6: **while** $p \leq n$ & $changelabel = 0$ **do**
7: SM$[cp_{p-1}, cp_p]$.
8: adjust the deviations as best as possible (using the CM).
9: update the network.
10: **if** the weights or existing points or their number are changed **then**
11: $startpoint \leftarrow \arg_p(cp_p)$ ($\arg_p(cp_p)$ is the value of p for which cp_p is the p^{th} control point).
12: $changelabel = 1$.
13: $n = n - 1$.
14: **else**
15: $p \leftarrow p + 1$
16: **end if**
17: **end while**
18: **end while**

end

3.2.2. Simulated annealing (SA)

Annealing is a thermal process in which a melt, initially at high temperature and disordered, is slowly cooled to make it reach a state of low energy. Simulated annealing is an analogous method for optimization. It is one of the local neighborhood search methods that allow '*uphill moves*' [15]. In other words, at high temperature, search is more likely to be random, thereby allowing worsening steps. However, there has to be some restriction on accepting such moves, otherwise the procedure would tend to search the whole solution space. Note that, as the process approaches zero temperature, the search takes the form of a pure greedy descent. The randomness helps the process to jump out of local minima. In this paper, we use a simple form of annealing to determine the timing of the project control points. The solution space of the simulated annealing consists of every n points among all the potential control points. To represent solution S, let $X = \{x_1, x_2, ..., x_m\}$ be the set of potential control points. Assuming that there are n control points, then S is a subset of X such that $|S| = n$. To build the initial solution, we use the *nearly equal interval* (NEI) strategy, i.e. we select the control points from among the potential control points such that the distance between any two control points is as close as possible to $[x_m/n]$ (where x_m is the last potential control point and n is the number of control points). For instance, if the number of control points is 5 and there are 11 potential control points at time instances 2, 5, 6, 8, 13, 15, 16, 20, 24, 31 and 32, then the control points are set at 6, 13, 20, 24 and 32. The objective function here is the project's finish time after executing the control process in the selected control points (i.e. making necessary adjustments at control points, the objective is to obtain a completion time for the project that is as near as possible to the baseline schedules completion time).

To create a neighborhood $s\prime$ of a solution s, we generate a random number r in $[1, n]$; then we substitute the r^{th} element of s by the right or the left point. Algorithm 4 explains this method in more detail. Inferior solutions are accepted with the probability $exp(-\delta/T)$ (δ is the increase in the objective function and T is a non-negative real-valued temperature parameter of the SA). In the SA, if the final solution is to be independent of the starting solution, the initial temperature T must be 'hot' enough to allow an almost free exchange of neighbouring solutions [15]. To set the SA parameters for the acceptance of 50% uphill moves, we initially solved all the instances of our experimental data set (see Section 4) with SA. The maximum variance observed in the SA solutions is around 7. Hence, the initial temperature is calculated as follows:

$$\exp(-7/T) = 0.5 \rightarrow T = -7/\ln(0.5) = 10.098.$$

So we start with $T = 10$ as the initial temperature. For the annealing schedule, we use a geometric cooling function, $\alpha(T) = aT$, where $a < 1$. Experience shows that relatively high values of a perform best and most reported successes in the literature use values between 0.8 and 0.99 [15]. Here, we use $a = 0.85$ and we reduce the temperature after every 5 iterations ($k_2 = 5$). The stopping condition is dependent on the number of solutions that are generated. Using this stopping condition, we studied the effect of parameters a and k_2 on the quality of the solutions. Our analysis showed a negligible effect. The only factor that seems to have some effects on the final solution is the number of generated solutions (k_1). We analyzed the effect of the number of generated solutions k_1 on the quality of the solutions in more detail. Our analysis is given in Subsection 4.2.1.

4. Computational experiments

In this section, we elaborate on the validation and performance of the FLM.

4.1. Validation of the FLM

To show the validity of the FLM and to evaluate its performance we compare its results with the results of the APC and the SA procedures. The consistency of the FLM results with those of the APC and the SA demonstrates that the FLM is an acceptable and valid control procedure. Moreover, due to the flexibility of the FLM in such aspects as the definition of the potential control points and the weights assigned to the activities as well as the potential control points makes it suitable for different projects with any type of workloads (front-loaded, end-loaded, etc).

Algorithm 4 Simulated Annealing Algorithm

Input:
the existing points (potential control points) $x_1, x_2, ..., x_m$;
the initial solution s_0 = equal intervals = $\{cp_1, ..., cp_n\}$;
the number of control points (n);
Output:
project finish time;
begin

1: $cp_0 \leftarrow 0, T \leftarrow 10, k_1 = k_2 = 0$
2: $sbest = s_0$
3: $p \leftarrow 1$
4: **while** $p \leq n$ **do**
5: SM$[cp_{p-1}, cp_p]$
6: adjust the deviations as best as possible (using the CM)
7: update the network
8: $p \leftarrow p + 1$
9: **end while**
10: $f_0 = C_{max}$ (C_{max} is the project finish time after executing loop 4-9).
11: $fbest = f_0$
12: **while** $k_1 <= 100$ **do**
13: $k_1 \leftarrow k_1 + 1$
14: $k_2 \leftarrow k_2 + 1$
15: $s_1 \leftarrow$ a new neighbourhood of s_0
16: $p \leftarrow 1$
17: **while** $p \leq n$ **do**
18: SM$[cp_{p-1}, cp_p]$
19: adjust the deviations as best as possible (using the CM)
20: update the network
21: $p \leftarrow p + 1$
22: **end while**
23: $f_1 = C_{max}$ (C_{max} is the project finish time after executing loop 17-22).
24: **if** $f_1 < f_0$ **then**
25: $s_0 = s_1$
26: $f_0 = f_1$
27: **else**
28: **if** $exp((f_0 - f_1)/T) > rand()$ **then**
29: $s_0 = s_1$
30: $f_0 = f_1$
31: **end if**
32: **end if**
33: **if** $f_1 < fbest$ **then**
34: $fbest = f_1$
35: $sbest = s_1$
36: **end if**
37: **if** $k_2 >= 5$ **then**
38: $T = 0.85 * T$
39: $k_2 = 0$
40: **end if**
41: **end while**

end

The benchmarks that are used for computational experiments are 48 instances from each of the well-known J30, J60, J90 and J120 sets of PSPLIB (J301_5, J302_5, ..., J3048_5, J601_5, J602_5, ..., J6048_5, J901_5, J902_5, ..., J9048_5, J1201_5, J1202_5, ..., J12048_5) [8].
The FLM, the APC and the SA have been coded in Matlab 2012 and the results have been obtained on a personal computer equipped with an Intel ® Core (TM) Duo CPU T2450 2.00 GHZ processor. Within the FLM, we utilize Cplex 12.3. All the statistical analysis has been performed using SPSS.
In this work, to obtain the activities' actual durations, we generate a symmetric triangular distribution in which the expected duration is equal to the deterministic PSPLIB activity duration and the left/right extensions are equal to 50% or 25% of the expected duration. In this way the possible late or early completion of activities occurs fairly equally. We name these two cases as Case 1 (left/right extension=50% of the expected duration) and Case 2 (left/right extension=25% of the expected duration). These two cases are used to have different possibilities for delays in the activities' durations. We set the shortest possible duration of each activity equal to the lowest value of its distribution.
Since at present we do not consider any limitation on the availability of resources, we use the earliest start time schedule as the baseline schedule.
Using a Student's t-distribution, we determine the appropriate number of simulation replications. For more details see Appendix B. After determining the appropriate number of simulation replications, we run the FLM and compute the relative delays in the projects' completions. Figures 1 and 2 display the mean relative delays in the projects' completions with respect to the number of control points used. With reference to Figure 1 and Figure 2, the appropriate numbers of control points (n) for our data sets are evident (regarding each data set, the corresponding curve steadies after a particular number of control points, i.e. no more improvement in the relative delays is observed when the number of controls are continued to be increased). For instance, for data set J60 no more improvement in the relative delays is observed by increasing the number of control points beyond 10 control points. Therefore, 10 is selected as the num-

ber of control points for the data set J60. Regarding the SA, we use the same number of control points for each instance of each data set. These values are shown in Table 2.

To compare the results obtained by the NEI (the

Figure 1: Effect of the number of control points on the relative delay for the FLM (Case 1)

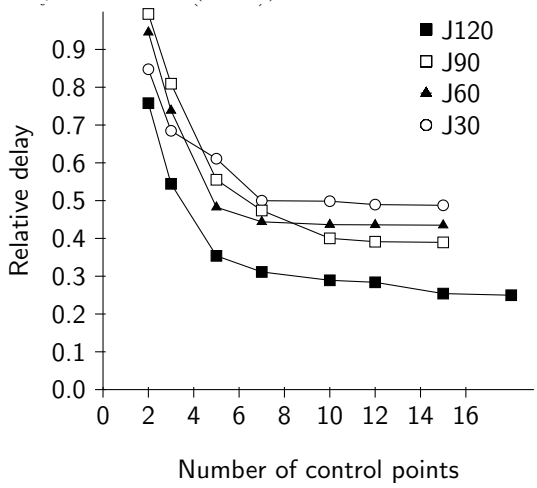


Table 2: Appropriate number of control points for different sets

Set	Number of control points (n)	
	Case 1	Case 2
J30	7	7
J60	10	10
J90	10	12
J120	15	15

initial solution for the SA), the SA, the FLM and the APC we compute the mean of the *relative delay percentage* of 48 instances from each data set. By relative delay we mean:
(actual project finish time-due date)/due date*100.
Actually, we use the percentage of relative delays as a quality metric to reflect the relation between the magnitude of the delay and the project's duration. It is obvious that 1 time unit delay for a project with a duration of 100 time units is not as important as the same delay for a project that has a duration of 5 time units.

The results for the sets J30, J60, J90 and J120

Figure 2: Effect of the number of control points on the relative delay for the FLM (Case 2)

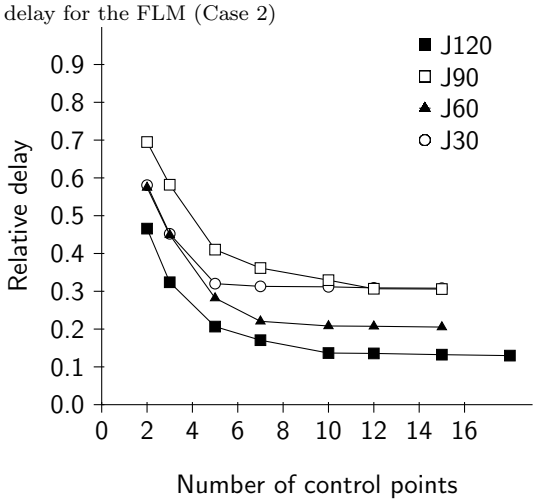


Table 3: The percentage of relative delay (Case 1)

Approach	J30	J60	J90	J120
NEI	1.06	0.74	0.89	0.43
SA	0.64	0.54	0.50	0.26
FLM	0.52	0.44	0.40	0.25
APC	0.46	0.37	0.38	0.24

are shown in Tables 3 and 4. We compare the results obtained by the FLM with the results of the SA and the APC to demonstrate the validity of the FLM. The *independent sample t-test* with $\alpha = 0.05$ is used to compare the obtained means and to investigate what is the difference between the FLM and alternative approaches. The p-values of these comparisons are shown in Table 5. As the SA searches a much larger number of solutions and the APC controls the project's progress at each and every existing control point, provided that the FLM results are not

Table 4: The percentage of relative delay (Case 2)

Approach	J30	J60	J90	J120
NEI	0.52	0.31	0.44	0.18
SA	0.37	0.24	0.35	0.15
FLM	0.31	0.21	0.31	0.13
APC	0.31	0.20	0.27	0.13

significantly different from those of the SA and the APC, it is indeed a demonstration of the validity of the FLM. For instance, with reference to Table 5, the performance of the FLM (when it determines the timing of control points and after adjusting the possible deviations) is compared to the performances of the SA and the APC. As it can be seen, the means of the relative delay percentages with respect to the FLM, the SA and the APC are not significantly different (for example, consider the p-values for the data set J30 in Case 1, when the FLM is compared with the SA (p-value=0.125) and when it is compared with the APC (p-value=0.436)).

Looking at the results in Tables 3 and 4, it can be seen that the relative delay in Case 2 is less than the relative delay in Case 1 for all instances. This is in line with the expectation, as the probability disruptions (delays) in Case 1 is higher than in Case 2. Random instances from the data sets J30, J60, J90 and J120 are tested to demonstrate the effectiveness of the FLM in controlling different types of projects having various number of activities. For the sake of brevity, we do not present the complete corresponding results in this paper. However, the complete results are available on line [21].

4.2. Performance of the FLM

4.2.1. The FLM versus the SA

In the SA (Algorithm 4), the stopping condition is based on the number of neighbors that are generated. In almost all the successful applications of the SA, reported in the literature, the best parameters are determined after much experimentation [15]. Accordingly, the number of generated solutions (k_1) are varied in order to study its effect on the quality of the SA solutions. Figure 3 shows the results for Case 1. A detailed exposition of this aspect of the SA is beyond the scope of this paper. However, as expected, with reference to Figure 3, it can be observed that the relative delays decrease (albeit with different rates of improvement) as the numbers of iterations increases. The FLM has less computational burden than the SA, because the number of vectors of solutions that the SA generates is much higher than the FLM. The CPU times with respect to the results of Figure 3 are presented in Figure 4. The rate of increase in the CPU times is considerably higher than the rate of improvement in the quality of solutions as a function of the number of generated solutions. Moreover, as is evident from Tables 3 and 4, the quality of the FLM solutions is very close to the results obtained by the APC. However, this is achieved with less computational time.

Table 6: The computational times for the FLM and the SA

Set	Computational time (sec)			
	Case 1		Case 2	
	FLM	SA	FLM	SA
J30	31.85	1705.30	29.16	2169.3
J60	128.69	6956.10	69.26	4614.3
J90	300.00	16126.5	265.69	11071.7
J120	566.18	12235.8	534.37	17556

Table 6 displays the computational times for the FLM and the SA. The SA consumes more CPU time than the FLM. For instance, for Case 1 and with respect to the J120 data set, the SA needs nearly 22 times more CPU time than the FLM. However, for Case 2, the SA needs nearly 33 times more CPU time than the FLM.

Figure 3: Effect of the number of SA iterations on the relative delay (Case 1)

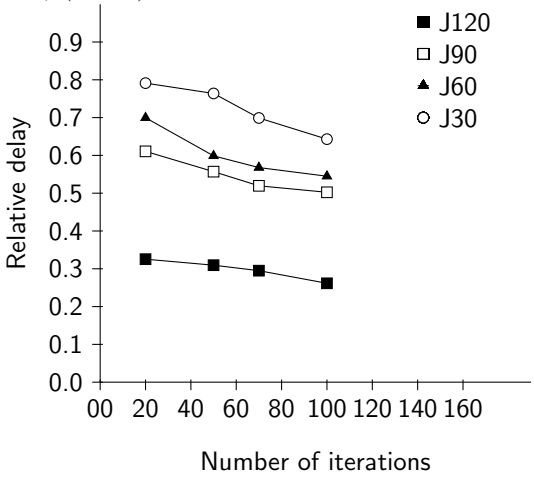
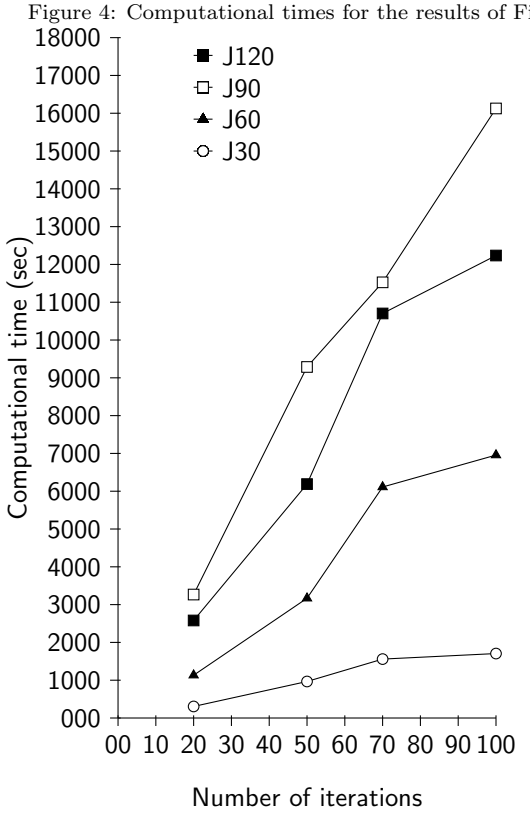


Table 5: SPSS results for comparison of the FLM with the SA and the APC

Approach	p-value							
	Case 1				Case 2			
	J30	J60	J90	J120	J30	J60	J90	J120
SA	0.125	0.200	0.339	0.887	0.399	0.567	0.693	0.494
APC	0.436	0.414	0.842	0.879	0.952	0.856	0.714	0.964

Figure 4: Computational times for the results of Figure 3



4.2.2. The FLM versus the APC

Note that in the FLM, the project is controlled at only a fraction of the potential control points, whereas in the APC the project is controlled at each potential control point. Table 7 shows the fraction of the potential control points that are used as control points in each data set. Considering that the process of controlling a project is costly, the results of Table 7 demonstrate the superiority of the FLM when compared to the APC results.

Table 7: Number of potential control points (pc points) considered by the FLM under Case 1 and Case 2

Set	No. of pc points	Considered no. of pc points		Considered % of pc points	
		Case 1	Case 2	Case 1	Case 2
J30	24	7	7	29	29
J60	41	10	10	24.5	24.5
J90	56	10	12	17.9	21.4
J120	64	15	15	23.4	23.4

4.2.3. The FLM versus the Raz and Erel approach

To the best of our knowledge, among those articles that are concerned with the determination of the timing of control points, Raz and ERel [14] utilize an analytical approach, namely dynamic programming to determine the timing of project control points under a static view (see Section 2). Their approach is based on maximizing the amount of information generated by the control points, which depends on the intensity of the activities, carried out since the last control point and on the time elapsed since their execution. The optimization problem is solved with a dynamic programming approach. They introduce two aspects that determine the amount of information generated by a control point: the activity intensity and the reporting delay [14]. In our approach, we use a different analytical model, namely a facility location model, to determine the timing of control points. It is interesting to compare the quality of the timing of control points obtained by these methods on the basis of relative delays. For so doing, we consider the reporting delay parameter (p) equal to 0.25, because according to the presented results in their paper, they obtained the best results when p=0.25. Raz and Erel explain in their paper that "the activity intensity can be any

measure that is used to define the project plan and in most cases changes according to time", so, to implement their method, we consider the activity intensity function $a(t)$ as follows (the idea is taken from De Falco and Macchiaroli [3]): Recall that m is the number of existing points (potential control points according to our definition); let NA_t $(t = 1, 2, ..., m)$ denote the number of activities that finish at point t according to the baseline schedule. To compute NA_t let:

$$a_{tj} = \begin{cases} 1 & \text{if } t-1 < sft(j) <= t \\ 0 & \text{otherwise} \end{cases}$$

where $sft(j)$ is the scheduled finish time of activity j. Now we define:

$$NA_t = \sum_{j=1}^{N} a_{tj} \tag{4.1}$$

and

$$TF_t = \sum_{j=1}^{N} a_{tj}.f_j \tag{4.2}$$

where f_j is the total float of activity j and N is the number of project activities. Now we consider $a(t)$ as follows:

$$a(t) = k\frac{NA_t}{(\sqrt{TF_t}+1)} \tag{4.3}$$

Constant k is defined such that:

$$\sum_{t=1}^{m} k\frac{NA_t}{(\sqrt{TF_t}+1)} = 100 \tag{4.4}$$

so,

$$a(t) = 100(\sum_{h=1}^{m} \frac{NA_h}{(\sqrt{TF_h}+1)})^{-1} \cdot \frac{NA_t}{(\sqrt{TF_t}+1)} \tag{4.5}$$

Now having determined the intensity function $a(t)$, we can proceed to implement Raz and Erel's method, i.e. finding the timing of the n control points by dynamic programming. Note that, in the FLM, we take $a(t)$ to be the weight assigned to point t. Given the two sets of the timings of control points obtained by the Raz and Erel's method and also by our method, and projecting possible disturbances via simulation and utilizing the CM for the possible adjustment of deviations, which set of timing leads to the project completion with the smallest percentage of relative delays? One more point to notice is that since Raz and Erel's method is static (the n control points are determined prior to the execution of the project) we ran the FLM in a static mode. We applied Raz and Erel's method on our data sets (48 instances from each of the J30, J60, J90 and J120 sets). The number of control points for each set is the same as in section 4.1. The results of this comparison for Case 1 are shown in Table 8. The second row of this table contains the mean of the percentage of relative delays in all data sets J30, J60, J90 and J120 when the control process is done in the control points that are obtained by Raz and Erel's method. The third row contains these results when the control process is done in the control points that are obtained by the FLM. The results for each data set have been compared by SPSS and the p-values are reported in the fourth row of Table 8. It shows that the FLM performs significantly better than Raz and Erel's method.

Table 8: Comparison of the Raz and Erel's method with the FLM (Case 1)

Approach	J30	J60	J90	J120
Raz and Erel	0.83	0.75	0.81	0.50
FLM	0.48	0.41	0.45	0.24
p-value	0.000	0.000	0.004	0.006

4.3. The CM performance

In the FLM, the SA, the APC and the Raz and Erel's method, and at each control point, Algorithm 2 is used to adjust the schedule, should the need arise. To demonstrate the effective role of the CM in our proposed methodology, all the test instances of Section 4.1 are run again, this time without the use of the CM. The corresponding results are reported in Table 9. With reference to Table 9 and comparing its results with those of Tables 3, 4 and 8, as expected, using the CM reduces the relative delay dramatically.

Table 10: Comparison of the presented CM and the classical crashing model (case 1)

Set	Percentage of the relative delay			
	J30	J60	J90	J120
FLM1	0.52	0.44	0.40	0.25
FLM2	0.87	0.71	0.84	0.46
Raz1	0.83	0.75	0.81	0.50
Raz2	1.08	1.01	1.10	0.66
p-value (FLM1 & FLM2)	0.000	0.004	0.000	0.003
p-value (Raz1 & Raz2)	0.004	0.025	0.031	0.125
p-value (FLM2 & Raz2)	0.030	0.011	0.050	0.031

For example, the average of the relative delay for instances in J30 in Table 3 after applying Algorithm 2 for reacting about the simulated delays is 0.52 and the corresponding value in Table 9 is 4.12, i.e., we could reduce the relative delay by 87.38%.

Another issue deserving further analysis is the strategy used in our implementation of the crashing model in selecting activities for crashing. In contrast to the classical time/cost trade-off problem, which selects activities for crashing based on their cost slopes, our proposed model selects activities based on their time distance to the current control point. In fact, an activity that: (1) has a scheduled start time greater than or equal to the current control point; (2) can still be crashed and (3) has a start time as near as possible to the present control point is selected (see Section 3.1.3). To compare our proposed CM with the classical crashing models, we have implemented the control process with a crashing model that is based on the activities' cost slope. The experimental results regarding the effectiveness of this strategy are listed in Table 10. We used the same benchmarks as in Section 4.1 for the experiments. However, the additional required data items, such as the activity normal and crashing costs, are generated randomly using uniform distributions. Four approaches are tested, the FLM where its CM is based on the proposed strategy (FLM1) (the same as in Table 3), the FLM where its CM is based on the cost slopes of the activities (a classical crashing model/FLM2), the Raz and Erel's method where its CM is based on the proposed strategy (Raz1) (the same as in Table 8) and finally the Raz and Erel method where its CM is based on the cost slopes of the activities (Raz2). In addition, we focused our attention on Case 1, where the probability of disruption is higher.

Table 9: The percentage of the relative delay without using the CM

Set	Percentage of the relative delay	
	Case 1	Case 2
J30	4.12	2.05
J60	4.06	1.85
J90	3.67	1.84
J120	3.49	1.67

As we expected, the proposed CM produces less relative delay when applied as an adjusting tool to repair the disrupted schedule. For instance, for the J30, the FLM1 yields approximately 40%, 37% and 52% less relative delay when compared to the FLM2, Raz1 and Raz2 methods, respectively. Considering the p-values, the significant difference between the methods performance is evident. To control projects with limited control budgets, one may be motivated to apply the crashing model where the criterion for selecting activities for crashing is their cost slopes. Table 10 shows that under this crashing strategy,the FLM performs better than the Raz and Erel's method. However, under limited control budget, a trade-off between the two crashing strategies should be made within the context of each project. This is investigated in the future work.

5. Conclusions and suggestions for future research

In this paper, we presented a dynamic control approach in which we used an adapted facility location model coupled with a computer simulation model and a project crashing model to dynamically determine the timing of control points in a project life cycle. We demonstrated the validity of our proposed method by comparing its results with the results obtained by two other algorithms that we designed for this purpose, namely all point control and simulated annealing. The analysis of the results regarding the performance of the FLM revealed its effectiveness for controlling the progress of different projects. It was shown that the FLM can dynamically control the standard PSPLIB instances when they are subjected to some random disruptions, and can complete them as near as possible to their original completion times. It was also shown that the FLM is flexible enough to determine the timing of control points using weight functions that define the important sections of any project. We also compared the performance of our method with that of another approach that utilizes an analytical method to determine the timing of the control points. The analysis of the results showed that our method yields a better timing of the control points which eventually leads to the disrupted project being completed with less delays.

Research on dynamic project control can take a number of directions in the future. For instance, scarce resources can be considered in the generation of the baseline schedule as well as its adjustments during the control process. In addition, as it is possible that at times the compression of activities is not sufficient to compensate for the projects delays, the implementation of proactive scheduling strategies such as the employment of different buffers can also be studied. Finally the trade-off between the projects delay penalties and the project control can also be studied.

Appendix A. Effect of the weight function on the timing of the control points

Consider the project that is shown in Figure 5. This is an activity on node network and the numbers in each node, from left to right, represent the activity's I.D. and its expected duration, respectively. Each activity duration is sampled from a symmetric triangular distribution in which the mean is equal to the expected duration and the left/right extensions are equal to 50% of the expected duration. The activity weights (ω_i) are shown above each node. After *forward pass computations*, the potential control points for this project are time points 1, 6, 7, 14, 20, 21, 22 and the corresponding a_js ((3.4)) are 1, 1, 0.5333, 0.5333, 0, 0, 0. If we use the weight function (3.2), the associated weights are 4.0156, 4.25, 4.2667, 6.6667, 0, 0, 0. After solving the FLM for 3 control points, the actual control points would be the time points 7, 14 and 22, resulting in a relative delay of 0.0193. As explained in section 3.1.1, the considered factors for determining the point weights in (3.2) are the activity weights as defined in (3.1) and the possible amount of compression from time instance j to the end of the project. Now suppose that we consider the latter one (possible amount of compression) as the only factor for defining the point weights, i.e. the weight function is changed and (3.3) is used to determine the weights of the potential control points. In this case, the associated weights are 3.75, 3.75, 2, 2, 0, 0, 0 and the actual control points in the first step would be the time points 1, 7 and 22, resulting in a relative delay of 0.0251. Note that 23.11% improvement is achieved when weight function (3.2) is used instead of weight function (3.3).

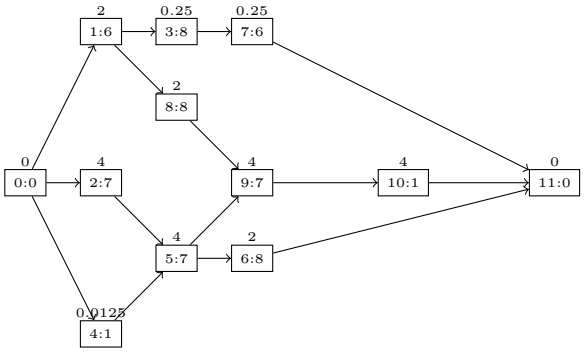


Figure 5: Project example

Appendix B. Determining the number of simulation replications

In order to determine the number of simulation replications, we used a Student's t-distribution. We initially ran an experiment with $nrep_0 = 30$ replications and determined $t_{nrep_0-1,1-\alpha/2}$ and observed $S_j^2(nrep_0)$ for each problem j in each set (J30, J60, J90, J120) and then determined

$$S^2(nrep_0) = max_j S_j^2(nrep_0)$$

Setting the target halfwidth $h = 0.5$ (suppose that we want our estimate of μ (mean) to be within 0.5 units of the true value), we obtain:

$$halfwidth\ h \simeq t_{nrep_0-1,1-\alpha/2}\sqrt{\frac{S^2(nrep_0)}{nrep}}$$

By solving it for $nrep$, we will have:

$$nrep = t^2_{nrep_0-1,1-\alpha/2}\frac{S^2(nrep_0)}{h^2}$$

so, for the data set J30, when there are 5 control points and $\alpha = 0.05$, 115 replications are required in Case 1. Similarly, with respect to the other sets, namely J60, J90 and J120, and for n=2, 3, 5, 7, 10, 12, 15 the number of required replications are obtained.

Acknowledgements

The authors would like to thank the reviewers for their invaluable comments on the paper and their detailed and constructive suggestions for improvement.

6. References

[1] R. A. Bowman, "Developing activity duration specification limits for effective project control", Eur J Oper Res, 174 (2006), 1191-1204.

[2] F. Deblaere, E. Demeulemeester, W. Herroelen, "Reactive scheduling in the multi-mode RCPSP", Computers and Operations Research, 38 (2011), 63-74.

[3] M. De Falco, R. Macchiaroli, "Timing of control activities in project planning", International Journal of Project Management, 16(1) (1998), 51-58.

[4] S. E. Elmaghraby, "On criticality and sensitivity in activity networks", Eur J Oper Res, 127 (2000), 220-238.

[5] D. Golenko-Ginzburg, Z. Laslo, "Timing control points via simulation for production systems under random disturbances", Mathematics and Computers in Simulation, 54 (2001), 451-458.

[6] W. Herroelen, R. Leus, "Project scheduling under uncertainty: Survey and research potentials", Eur J Oper Res, 165 (2005), 289 - 306.

[7] K. Kogan, T. Raz, R. Elitzur, "Optimal control in homogeneous projects: analytically solvable deterministic cases, IIE Transaction, 34(1) (2002), 63-75.

[8] R. Kolisch, A. Sprecher, "PSPLIB - A project scheduling library", Eur J Oper Res, 96 (1997), 205-216.

[9] J. Kuster, D. Jannach, G. Friedrich, "Extending the RCPSP for modeling and solving disruption management problems", Appl Intell, 31 (2009), 234 - 253.

[10] A. Law, W. Kelton, "Simulation modelling and analysis", Third edition, McGraw-Hill series,(2000).

[11] B. Mirchandani, R. L. Francis, "Discrete Location Theory", John Wiley & Sons Inc. New York, (1989).

[12] A. Oudjit, "Median Locations on Deterministic and Probabilistic Multidimentional Networks", Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, New York.

[13] F.Y. Partovi, J. Burton, "Timing of monitoring and control of CPM projects", IEEE Transactions on Engineering Management, 40(1) (1993), 68-75.

[14] T. Raz, E. Erel, "Optimal timing of project control points", Eur J Oper Res, 127 (2000), 252 - 261.

[15] C. R. Reeves, "Modern heuristic techniques for combinatorial problems", Jonh Wiley & sons, Inc. New York, (1993), 20-69.

[16] H.R. Tareghian, M. Salari, "On the optimal frequency and timing of control points in a project's life cycle", International Journal of Industrial Engineering and Production Research, 20(3)(2009), 92 - 98.

[17] S. Van de Vonder, E. Demeulemeester, W. Herroelen, "A classification of predictive-reactive project scheduling procedures", J Sched, 10 (2007), 195 - 207.

[18] M. Vanhoucke, "On the dynamic use of project performance and schedule risk information during project tracking", Omega, 39 (2011), 416-426.

[19] G. Zhu, J.F. Bard, G. Yu, "Disruption management for resource-constrained project scheduling", J Oper Res Soc 56 (2005), 365 - 381.

[20] http://www.projectcontrolsonline.com/Home /DefinitionandImportanceofProjectControls.aspx.

[21] http://sabeghi.student.um.ac.ir/index.php? option=com_content&view=article&id=26635& Itemid=60866&limitstart=2.